Project 1

This project is going to include some “simulation”, since most of you do not know about networks. This project will simulate and simplify a bit what happens when your webbrowser contacts a webserver to download a webpage using the HTTP/TLS protocol.

You will act as a client and on another account will act as a server.

Submit in **pdf**:

- A brief writeup of what you did. A flowchart or a picture can make this clearer rather than text only.
- List of commands, along with screenshots, of what you did.

More details:

- You don’t need to write a program to do any of this; you can type each command one at a time like we did in previous labs.
- When I say “send”, there is no “sending” over the network. Use /usr/local/src/project1-<username> to “send” the message. As an analogy, let’s say you are the server. You put (“write”) the message on your window. At some point, the client will walk by and see (“read”) the message. The client can then put (“write”) their own message on their window. At some point the server will walk by and see (“read”) the message. And so on.

Assume there is a Trusted Third Party (TTP), a client C, and a server S.

Setup

1) Create a public/private key pair for the TTP.
2) Import public key of TTP for C.
3) Create public/private key pair for S.
4) Have TTP sign the public key for S.

C now wants to connect to S.

Handshake

5) C “downloads” signed public key of S
6) Verify the public key’s signature is correct.
7) C encrypts “Hello” using S’s public key

8) S decrypts the message and sends back a list of supported algorithms (e.g. AES-128-CBC, etc.)

You can pick the list of algorithms

9) C selects one of these algorithms, generates a secret key, and sends that secret key and selected algorithm, encrypted under S's public key.
10) C encrypts "Ready" using the secret key and creates a hash.
11) C sends both the encrypted message and hash to S.
12) S verifies integrity of message and decrypts the message.
13) S encrypts "Ready 2!" using the secret key and creates a hash.
11) S sends both the encrypted message and hash to C.
12) C verifies integrity of message and decrypts the message.

Reminder: When I say "encrypts" and "sends", you can encrypt a file and then write it to your public project1 folder. E.g. if your username is hill, your public folder is /usr/local/src/project1-hill.